



RRRRRRRR	MM	MM	11	IIIIII	NN	NN	PPPPPPPP	SSSSSSSS	CCCCCCCC	NN	NN	
RRRRRRRR	MM	MM	11	IIIIII	NN	NN	PPPPPPPP	SSSSSSSS	CCCCCCCC	NN	NN	
RR	RR	MMMM	MMMM	1111	II	NN	PP	PP	CC	NN	NN	
RR	RR	MMMM	MMMM	1111	II	NN	PP	PP	CC	NN	NN	
RR	RR	MM	MM	11	II	NNNN	PP	PP	CC	NNNN	NN	
RR	RR	MM	MM	11	II	NNNN	PP	PP	CC	NNNN	NN	
RRRRRRRR	MM	MM	11	II	NN	NN	PPPPPPPP	SSSSSS	CC	NN	NN	
RRRRRRRR	MM	MM	11	II	NN	NN	PPPPPPPP	SSSSSS	CC	NN	NN	
RR	RR	MM	MM	11	II	NN	NNNN	PP	CC	NN	NNNN	
RR	RR	MM	MM	11	II	NN	NNNN	PP	CC	NN	NNNN	
RR	RR	MM	MM	11	II	NN	NN	PP	SS	CC	NN	NN
RR	RR	MM	MM	11	II	NN	NN	PP	SS	CC	NN	NN
RR	RR	MM	MM	111111	IIIIII	NN	NN	PP	SSSSSS	CCCCCCCC	NN	NN
RR	RR	MM	MM	111111	IIIIII	NN	NN	PP	SSSSSS	CCCCCCCC	NN	NN

LL	IIIIII	SSSSSSSS
LL	IIIIII	SSSSSSSS
LL	II	SS
LLLLLLLLLL	IIIIII	SSSSSSSS
LLLLLLLLLL	IIIIII	SSSSSSSS

(2)	65	DECLARATIONS
(3)	109	RM\$INPUT_SCAN - CHECK SYSSINPUT FOR \$, \$EOD, OR \$DECK RECORD
(15)	465	DCL_SCAN-SUBROUTINE

0000 1 \$BEGIN RM1INPSCN,000,RMSRMS1,<SYSS\$INPUT \$, SEOD, & SDECK ROUTINES>  
0000 2  
0000 3  
0000 4 \*\*\*\*\*  
0000 5 \*  
0000 6 \* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
0000 7 \* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
0000 8 \* ALL RIGHTS RESERVED.  
0000 9 \*  
0000 10 \* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
0000 11 \* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
0000 12 \* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
0000 13 \* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
0000 14 \* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
0000 15 \* TRANSFERRED.  
0000 16 \*  
0000 17 \* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
0000 18 \* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
0000 19 \* CORPORATION.  
0000 20 \*  
0000 21 \* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
0000 22 \* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
0000 23 \*  
0000 24 \*  
0000 25 \*\*\*\*\*  
0000 26  
0000 27 ++  
0000 28 Facility: rms32  
0000 29  
0000 30 Abstract:  
0000 31 this module performs end of file checking, as well as  
0000 32 \$deck processing for \$get /\$find on sys\$input  
0000 33 processing for the sequential file organization.  
0000 34  
0000 35 Environment:  
0000 36 star processor running starlet exec.  
0000 37  
0000 38 Author: L F Laverdure, creation date: 29-JAN-1978  
0000 39  
0000 40 Modified By:  
0000 41  
0000 42 V03-002 KBT0140 Keith B. Thompson 20-Aug-1982  
0000 43 Reorganize psects  
0000 44  
0000 45 V03-001 KBT0085 Keith B. Thompson 13-Jul-1982  
0000 46 Clean up some of the psect nonsense  
0000 47  
0000 48 V02-006 RAS0017 Ron Schaefer 21-Jul-1981  
0000 49 Change buffer management so it can work with stream files.  
0000 50 In particular, this routine only scans the current buffer,  
0000 51 the caller must provide an adequate buffer size.  
0000 52  
0000 53 V005 REFORMAT Ken Henderson 30-JUL-1980 6:24  
0000 54 the code was reformatted  
0000 55  
0000 56  
0000 57 Revision History:

RM1INPSCN  
V04-000

SYSS\$INPUT \$, \$EDD, & \$DECK ROUTINES

E 15

16-SEP-1984 00:49:27 VAX/VMS Macro V04-00  
5-SEP-1984 16:23:26 [RMS.SRC]RM1INPSCN.MAR;1

Page 2  
(1)

0000	58	:	
0000	59	:	
0000	60	:	01
0000	61	:	--
0000	62	:	
0000	63	:	

L F Laverdure, 17-JUN-1978 20:57

```
0000 65      .SBTTL DECLARATIONS
0000 66
0000 67
0000 68 ; Include Files:
0000 69 ;
0000 70
0000 71 ;
0000 72 ; Macros:
0000 73 ;
0000 74
0000 75     $IRBDEF
0000 76     $IFBDEF
0000 77     $DEVDEF
0000 78     $BDBDEF
0000 79     $PIODEF
0000 80     $RMSDEF
0000 81
0000 82 ;
0000 83 ; macro to call dcl_scan subroutine, setting up in-line argument list
0000 84 ; of match string and equality branch offset
0000 85 ;
0000 86
0000 87     .MACRO DCL_SCAN STR=,EQUAL=,DISP=W,?L
0000 88     BSB'DISP      DCL_SCAN
0000 89     .ASCIC %STR%
0000 90 L:    .BYTE EQUAL-L
0000 91     .ENDM DCL_SCAN
0000 92
0000 93 ;
0000 94 ; Equated Symbols:
0000 95 ;
0000 96
0000 97     TAB      = 9      ; horizontal tab
0000 98     LOWERCASE_A    = 97    ; lower case a
0000 99     LOWERCASE_Z    = 122   ; lower case z
0000 100
0000 101    EODSTR_MAXLEN = 15    ; max. len. of end-of-data scan string
0000 102
0000 103 ;
0000 104 ;
0000 105 ; Own Storage:
0000 106 ;
0000 107
```

0000 109 .SBTTL RM\$INPUT\_SCAN - CHECK SYSSINPUT FOR \$, \$EOD, OR \$DECK RECORD  
0000 110  
0000 111 :++  
0000 112 RMSINPUT\_SCAN  
0000 113  
0000 114 rm\$input\_scan routine to check the current (non-terminal) record for  
0000 115 matching the current sys\$input end-of-data scan string. this will be  
0000 116 either a single '\$' or some user-defined string. if the record matches  
0000 117 a user-defined string, cause this record to be skipped and return a  
0000 118 single rms\$eof error (i.e., do not latch), allowing further reads to  
0000 119 access subsequent records.  
0000 120  
0000 121 if matching a single '\$', check if the pio\$v\_eod flag is on, specifying  
0000 122 that only a match of '\$eod' is to be scanned for. if so, then proceed as  
0000 123 for a match of a user-defined string above if the record contains \$eod.  
0000 124  
0000 125 if not matching \$eod, try for a match on \$deck, and if so perform appropriate  
0000 126 processing. if the record is not \$deck, (i.e., it is some other record  
0000 127 beginning with a '\$'), return rms\$eof error and do not skip this record  
0000 128 so that subsequent \$gets or \$finds by the user will also encounter this  
0000 129 eof record.  
0000 130  
0000 131 if the record matches none of the above cases, simply return. the record  
0000 132 will be processed normally, that is, it will be gotten or found for the user.  
0000 133  
0000 134 Input Parameters:  
0000 135  
0000 136 r11 impure area addr  
0000 137 r10 ifab addr  
0000 138 r9 irab addr  
0000 139 r8 rab addr  
0000 140 r7 addr of end of buffer+1  
0000 141 r6 size of record in bytes  
0000 142 r1 addr of record in the buffer  
0000 143  
0000 144 Implicit Inputs:  
0000 145  
0000 146 the contents of the various structures.  
0000 147 pio\$gt\_endstr the current end-of-data scan string  
0000 148 pio\$v\_eod \$eod flag  
0000 149 note: if sys\$input is from a disk file and records are allowed to  
0000 150 cross block boundaries, irb\$b\_mbc must be greater than 0,  
0000 151 otherwise the processing of a record crossing a block boundary  
0000 152 will cause rms to loop. it is assumed that rm\$connect1 has  
0000 153 forced mbc > 0.  
0000 154  
0000 155 (note: it has already been determined externally to this routine that  
0000 156 this is the sys\$input stream, indirectly accessed, on a non-terminal device.  
0000 157  
0000 158 outputs:  
0000 159  
0000 160 r2-r5,ap destroyed  
0000 161 r0 status code  
0000 162  
0000 163 Implicit Outputs:  
0000 164  
0000 165 : irb\$1\_curbdb may be cleared

0000 166 : irb\$v\_find set if record is to be skipped (\$eod, user-defined eod, or \$deck)  
0000 167 : irb\$v\_ppf\_eof set if \$eod or user-defined eod. after record is skipped, flags return of rms\$\_eof status. in this case return status from rm\$input\_scan is rms\$\_suc.  
0000 168 : irb\$v\_ppf\_skip set if record is \$deck. causes next record to be processed after \$deck record is skipped.  
0000 169 : irb\$v\_ppf\_fndsv saves original state of irb\$v\_find when irb\$v\_ppf\_skip is set  
0000 170 :  
0000 171 :  
0000 172 :  
0000 173 :  
0000 174 :  
0000 175 :  
0000 176 : completion code:  
0000 177 :  
0000 178 : standard rms (suc or eof), except that a special code of hex 10000 will be returned in r0 to indicate that record to be matched was not entirely contained within the buffer. in this case the current bdb has been released and the buffer need merely be refilled.  
0000 179 :  
0000 180 :  
0000 181 :  
0000 182 :  
0000 183 : Side Effects:  
0000 184 :  
0000 185 : matching the eod string resets the eod string to a single '\$'.  
0000 186 : \$deck may set this to something else.  
0000 187 : invalid syntax on '\$' records will cause a hard eof, which will cause the cli to process the offending record.  
0000 188 :  
0000 189 :  
0000 190 :--  
0000 191 :  
0000 192 :  
0000 193 :  
0000 194 :  
0000 195 :  
0000 196 :  
0000 197 :  
0000 198 :  
0000 199 :  
0000 200 :  
0000 201 :  
0000 202 :  
0000 203 :  
0000 204 :  
0000 205 :  
0000 206 :  
0000 207 :  
0000 208 :  
0000 209 :  
0000 210 :  
0000 211 :  
0000 212 :  
0000 213 :  
0000 214 :  
0000 215 :  
0000 216 :  
0000 217 :  
0000 218 :  
0000 219 :  
0000 220 :  
0000 221 :  
0000 222 :  
0000 223 :  
0000 224 :  
0000 225 :  
0000 226 :  
0000 227 :  
0000 228 :  
0000 229 :  
0000 230 :  
0000 231 :  
0000 232 :  
0000 233 :  
0000 234 :  
0000 235 :  
0000 236 :  
0000 237 :  
0000 238 :  
0000 239 :  
0000 240 :  
0000 241 :  
0000 242 :  
0000 243 :  
0000 244 :  
0000 245 :  
0000 246 :  
0000 247 :  
0000 248 :  
0000 249 :  
0000 250 :  
0000 251 :  
0000 252 :  
0000 253 :  
0000 254 :  
0000 255 :  
0000 256 :  
0000 257 :  
0000 258 :  
0000 259 :  
0000 260 :  
0000 261 :  
0000 262 :  
0000 263 :  
0000 264 :  
0000 265 :  
0000 266 :  
0000 267 :  
0000 268 :  
0000 269 :  
0000 270 :  
0000 271 :  
0000 272 :  
0000 273 :  
0000 274 :  
0000 275 :  
0000 276 :  
0000 277 :  
0000 278 :  
0000 279 :  
0000 280 :  
0000 281 :  
0000 282 :  
0000 283 :  
0000 284 :  
0000 285 :  
0000 286 :  
0000 287 :  
0000 288 :  
0000 289 :  
0000 290 :  
0000 291 :  
0000 292 :  
0000 293 :  
0000 294 :  
0000 295 :  
0000 296 :  
0000 297 :  
0000 298 :  
0000 299 :  
0000 300 :  
0000 301 :  
0000 302 :  
0000 303 :  
0000 304 :  
0000 305 :  
0000 306 :  
0000 307 :  
0000 308 :  
0000 309 :  
0000 310 :  
0000 311 :  
0000 312 :  
0000 313 :  
0000 314 :  
0000 315 :  
0000 316 :  
0000 317 :  
0000 318 :  
0000 319 :  
0000 320 :  
0000 321 :  
0000 322 :  
0000 323 :  
0000 324 :  
0000 325 :  
0000 326 :  
0000 327 :  
0000 328 :  
0000 329 :  
0000 330 :  
0000 331 :  
0000 332 :  
0000 333 :  
0000 334 :  
0000 335 :  
0000 336 :  
0000 337 :  
0000 338 :  
0000 339 :  
0000 340 :  
0000 341 :  
0000 342 :  
0000 343 :  
0000 344 :  
0000 345 :  
0000 346 :  
0000 347 :  
0000 348 :  
0000 349 :  
0000 350 :  
0000 351 :  
0000 352 :  
0000 353 :  
0000 354 :  
0000 355 :  
0000 356 :  
0000 357 :  
0000 358 :  
0000 359 :  
0000 360 :  
0000 361 :  
0000 362 :  
0000 363 :  
0000 364 :  
0000 365 :  
0000 366 :  
0000 367 :  
0000 368 :  
0000 369 :  
0000 370 :  
0000 371 :  
0000 372 :  
0000 373 :  
0000 374 :  
0000 375 :  
0000 376 :  
0000 377 :  
0000 378 :  
0000 379 :  
0000 380 :  
0000 381 :  
0000 382 :  
0000 383 :  
0000 384 :  
0000 385 :  
0000 386 :  
0000 387 :  
0000 388 :  
0000 389 :  
0000 390 :  
0000 391 :  
0000 392 :  
0000 393 :  
0000 394 :  
0000 395 :  
0000 396 :  
0000 397 :  
0000 398 :  
0000 399 :  
0000 400 :  
0000 401 :  
0000 402 :  
0000 403 :  
0000 404 :  
0000 405 :  
0000 406 :  
0000 407 :  
0000 408 :  
0000 409 :  
0000 410 :  
0000 411 :  
0000 412 :  
0000 413 :  
0000 414 :  
0000 415 :  
0000 416 :  
0000 417 :  
0000 418 :  
0000 419 :  
0000 420 :  
0000 421 :  
0000 422 :  
0000 423 :  
0000 424 :  
0000 425 :  
0000 426 :  
0000 427 :  
0000 428 :  
0000 429 :  
0000 430 :  
0000 431 :  
0000 432 :  
0000 433 :  
0000 434 :  
0000 435 :  
0000 436 :  
0000 437 :  
0000 438 :  
0000 439 :  
0000 440 :  
0000 441 :  
0000 442 :  
0000 443 :  
0000 444 :  
0000 445 :  
0000 446 :  
0000 447 :  
0000 448 :  
0000 449 :  
0000 450 :  
0000 451 :  
0000 452 :  
0000 453 :  
0000 454 :  
0000 455 :  
0000 456 :  
0000 457 :  
0000 458 :  
0000 459 :  
0000 460 :  
0000 461 :  
0000 462 :  
0000 463 :  
0000 464 :  
0000 465 :  
0000 466 :  
0000 467 :  
0000 468 :  
0000 469 :  
0000 470 :  
0000 471 :  
0000 472 :  
0000 473 :  
0000 474 :  
0000 475 :  
0000 476 :  
0000 477 :  
0000 478 :  
0000 479 :  
0000 480 :  
0000 481 :  
0000 482 :  
0000 483 :  
0000 484 :  
0000 485 :  
0000 486 :  
0000 487 :  
0000 488 :  
0000 489 :  
0000 490 :  
0000 491 :  
0000 492 :  
0000 493 :  
0000 494 :  
0000 495 :  
0000 496 :  
0000 497 :  
0000 498 :  
0000 499 :  
0000 500 :  
0000 501 :  
0000 502 :  
0000 503 :  
0000 504 :  
0000 505 :  
0000 506 :  
0000 507 :  
0000 508 :  
0000 509 :  
0000 510 :  
0000 511 :  
0000 512 :  
0000 513 :  
0000 514 :  
0000 515 :  
0000 516 :  
0000 517 :  
0000 518 :  
0000 519 :  
0000 520 :  
0000 521 :  
0000 522 :  
0000 523 :  
0000 524 :  
0000 525 :  
0000 526 :  
0000 527 :  
0000 528 :  
0000 529 :  
0000 530 :  
0000 531 :  
0000 532 :  
0000 533 :  
0000 534 :  
0000 535 :  
0000 536 :  
0000 537 :  
0000 538 :  
0000 539 :  
0000 540 :  
0000 541 :  
0000 542 :  
0000 543 :  
0000 544 :  
0000 545 :  
0000 546 :  
0000 547 :  
0000 548 :  
0000 549 :  
0000 550 :  
0000 551 :  
0000 552 :  
0000 553 :  
0000 554 :  
0000 555 :  
0000 556 :  
0000 557 :  
0000 558 :  
0000 559 :  
0000 560 :  
0000 561 :  
0000 562 :  
0000 563 :  
0000 564 :  
0000 565 :  
0000 566 :  
0000 567 :  
0000 568 :  
0000 569 :  
0000 570 :  
0000 571 :  
0000 572 :  
0000 573 :  
0000 574 :  
0000 575 :  
0000 576 :  
0000 577 :  
0000 578 :  
0000 579 :  
0000 580 :  
0000 581 :  
0000 582 :  
0000 583 :  
0000 584 :  
0000 585 :  
0000 586 :  
0000 587 :  
0000 588 :  
0000 589 :  
0000 590 :  
0000 591 :  
0000 592 :  
0000 593 :  
0000 594 :  
0000 595 :  
0000 596 :  
0000 597 :  
0000 598 :  
0000 599 :  
0000 600 :  
0000 601 :  
0000 602 :  
0000 603 :  
0000 604 :  
0000 605 :  
0000 606 :  
0000 607 :  
0000 608 :  
0000 609 :  
0000 610 :  
0000 611 :  
0000 612 :  
0000 613 :  
0000 614 :  
0000 615 :  
0000 616 :  
0000 617 :  
0000 618 :  
0000 619 :  
0000 620 :  
0000 621 :  
0000 622 :  
0000 623 :  
0000 624 :  
0000 625 :  
0000 626 :  
0000 627 :  
0000 628 :  
0000 629 :  
0000 630 :  
0000 631 :  
0000 632 :  
0000 633 :  
0000 634 :  
0000 635 :  
0000 636 :  
0000 637 :  
0000 638 :  
0000 639 :  
0000 640 :  
0000 641 :  
0000 642 :  
0000 643 :  
0000 644 :  
0000 645 :  
0000 646 :  
0000 647 :  
0000 648 :  
0000 649 :  
0000 650 :  
0000 651 :  
0000 652 :  
0000 653 :  
0000 654 :  
0000 655 :  
0000 656 :  
0000 657 :  
0000 658 :  
0000 659 :  
0000 660 :  
0000 661 :  
0000 662 :  
0000 663 :  
0000 664 :  
0000 665 :  
0000 666 :  
0000 667 :  
0000 668 :  
0000 669 :  
0000 670 :  
0000 671 :  
0000 672 :  
0000 673 :  
0000 674 :  
0000 675 :  
0000 676 :  
0000 677 :  
0000 678 :  
0000 679 :  
0000 680 :  
0000 681 :  
0000 682 :  
0000 683 :  
0000 684 :  
0000 685 :  
0000 686 :  
0000 687 :  
0000 688 :  
0000 689 :  
0000 690 :  
0000 691 :  
0000 692 :  
0000 693 :  
0000 694 :  
0000 695 :  
0000 696 :  
0000 697 :  
0000 698 :  
0000 699 :  
0000 700 :  
0000 701 :  
0000 702 :  
0000 703 :  
0000 704 :  
0000 705 :  
0000 706 :  
0000 707 :  
0000 708 :  
0000 709 :  
0000 710 :  
0000 711 :  
0000 712 :  
0000 713 :  
0000 714 :  
0000 715 :  
0000 716 :  
0000 717 :  
0000 718 :  
0000 719 :  
0000 720 :  
0000 721 :  
0000 722 :  
0000 723 :  
0000 724 :  
0000 725 :  
0000 726 :  
0000 727 :  
0000 728 :  
0000 729 :  
0000 730 :  
0000 731 :  
0000 732 :  
0000 733 :  
0000 734 :  
0000 735 :  
0000 736 :  
0000 737 :  
0000 738 :  
0000 739 :  
0000 740 :  
0000 741 :  
0000 742 :  
0000 743 :  
0000 744 :  
0000 745 :  
0000 746 :  
0000 747 :  
0000 748 :  
0000 749 :  
0000 750 :  
0000 751 :  
0000 752 :  
0000 753 :  
0000 754 :  
0000 755 :  
0000 756 :  
0000 757 :  
0000 758 :  
0000 759 :  
0000 760 :  
0000 761 :  
0000 762 :  
0000 763 :  
0000 764 :  
0000 765 :  
0000 766 :  
0000 767 :  
0000 768 :  
0000 7

0000 193 RMSINPUT\_SCAN::  
0000 194  
0000 195 :  
0000 196 : Assume entire record is contained in the buffer.  
51 DD 0000 197 :  
0002 198 PUSHL R1 ; save the current record offset  
0002 199  
0002 200 :++  
0002 201 :  
0002 202 now try to match against end of data string  
0002 203 :  
0002 204 :--  
0002 205 :  
53 00000000'9F 9E 0002 206 MOVAB @#PIO\$GT\_ENDSTR,R3 ; get addr of end of data string  
52 83 9A 0009 207 MOVZBL (R3)+,R2 ; get string length  
52 56 B1 000C 208 CMPW R6,R2 ; is record at least this long?  
06 1F 000F 209 BLSSU SU\$XIT ;  
63 61 52 29 0011 210 CMPC3 R2,(R1),(R3) ; no = no match  
06 13 0015 211 BEQL EOD\_MATCH ; eod string match?  
0017 212 : branch if yes  
0017 213 :  
0017 214 : no - simply return  
0017 215 :  
0017 216 :  
0017 217 SU\$XIT: RMSSUC  
02 BA 001A 218 SCNXIT: POPR #^M<R1> ; restore record address  
05 001C 219 SCNRSB: RSB

001D 221  
001D 222 :++  
001D 223 : have matched the current end-of-data string  
001D 224 : if matching a single '\$', must check for \$eod and \$deck, else have  
001D 225 : matched a user-defined (via \$deck) string and must skip it, giving  
001D 226 : a single eof error.  
001D 227 :--  
001D 230 :  
001D 231 :  
001D 232 EOD\_MATCH:  
2401 8F 00000000'9F B1 001D 233 CMPW @#PIO\$GT\_ENDSTR,#1+<^A/\$/08> ; are we matching single '\$'?  
29 12 0026 001D 234 BNEQ END\_OF\_DATA ; branch if not  
50 56 01 C3 0028 001D 235 SUBL3 #1,R6,R0 ; set remaining byte count  
002C 0028 001D 236 DCL\_SCAN <EOD>,EQUAL=END\_OF\_DATA1 ; scan for 'eod'  
0034 0034 001D 237  
0034 0034 001D 238 : record not \$eod  
0034 0034 001D 239 :  
0034 0034 001D 240 :  
DB 00000000'9F 01 E0 0034 241 BBS #PIO\$V\_EOD,@#PIO\$GW\_STATUS,SUCXIT ; branch if only matching '\$eod'  
003C 0034 001D 242 DCL\_SCAN <DECK>,EQUAL=GOT\_DECK ; scan for 'deck'  
0045 0045 001D 243  
0045 0045 001D 244 :++  
0045 0045 001D 245 : have found a '\$' record that is neither \$eod nor \$deck.  
0045 0045 001D 246 :  
0045 0045 001D 247 : return rms\$eof and reset the nrp to find this record again.  
0045 0045 001D 248 :  
0045 0045 001D 249 :--  
0045 0045 001D 250 :  
0045 0045 001D 251 :  
0045 0045 001D 252 :--  
40 A9 48 A9 7D 004A 0045 253 SETEOF: RMSEOF EOF  
C9 11 004F 0045 254 MOVQ IRBSL\_RP\_VBN(R9),IRBSL\_NRP\_VBN(R9)  
0045 255 BRB SCNxit

00000000'9F 2401 8F B0 0B 11 0051 257  
0051 258 ;++  
0051 259 : have encounter user-defined end-of-data string  
0051 260 : reset end-of-data scan string to match a single '\$'  
0051 261  
0051 262  
0051 263  
0051 264 ;--  
0051 265  
0051 266 END\_OF\_DATA:  
0051 267 MOVW #1+<^A/\$/08>, @#PIO\$GT-ENDSTR  
005A 268 BRB EOD1  
005C 269  
005C 270 ;++  
005C 271 : have encounter \$eod  
005C 272  
005C 273 : clear irb\$V\_ppf\_eod so that any '\$' record will give eof error, and  
005C 274 : set irb\$V\_ppf\_eof to cause eof to be returned after record has been skipped.  
005C 275  
005C 276  
005C 277 ;--  
005C 278  
005C 279 END\_OF\_DATA1:  
005C 280 BICB2 #1@PIO\$V\_EOD, @#PIO\$GW\_STATUS; clear eod flag  
50 D5 0063 281 TSTL R0 ; any other tokens seen?  
DE 12 0065 282 BNEQ SETEOF ; branch if yes (error)  
0067 283 EOD1: SSB #IRB\$V\_PPF\_EOF, (R9) ; cause eof to be returned  
006B 284 SET\_FIND:  
006B 285 SSB #IRB\$V\_FIND, (R9) ; set find bit to skip record  
A6 11 006F 286 SUC\_BR: BRB SUCXIT

0071 288  
0071 289 :++  
0071 290 : have found \$deck  
0071 292 : scan for /dollars qualifier  
0071 294  
0071 295 :--  
0071 296  
0071 297 GOT\_DECK:  
C8 12 007B 298 DCL SCAN </DOLL>,EQUAL=GOT\_DOLLARS ; scan for '/doll'  
007D 300 BNEQ SETEOF ; branch if something other than  
007D 301  
007D 302 : '/doll' seen (error)  
007D 303 :++  
007D 304 : saw \$deck, either with no qualifier or with /dollars and either a null or  
007D 305 : no argument.  
007D 306 : in any case, set pio\$v\_eod to indicate '\$eod' is the logical end-of-data string  
007D 307  
007D 308  
007D 309  
007D 310 :--  
007D 311  
007D 312  
00000000'9F 02 88 007D 313 SETEOD: BISB2 #1@PIO\$V\_EOD,0#PIO\$GW\_STATUS  
0084 314 SET\_SKIP:  
0084 315 SSB #IRB\$V\_PPF\_SKIP,(R9) ; set flag to skip \$deck record  
0088 316  
0088 317  
0088 318 : and read the next  
0088 319  
0088 320  
DF 69 29 E1 0088 321 BBC #IRB\$V\_FIND,(R9),SET FIND; branch if doing \$get  
DF 69 30 E3 008C 322 BBCS #IRB\$V\_PPF\_FNDSV,(R9),SUC\_BR; save find bit and branch

```

0090 324 :++
0090 325 : have found $deck /dollars
0090 326 : scan for end-of-data string value indicator ('=' or ':')
0090 327 :--+
0090 328 : GOT_DOLLARS:
0090 329 : DCL_SCAN <:,>,EQUAL=GOT_ARG ; scan for ':'
0096 330 :--+
E5 13 0096 331 : BEQL SETEOD ; branch if nothing else in record
0098 332 : DCL_SCAN <=,>,EQUAL=GOT_ARG ; scan for '='
009E 333 :--+
A5 11 009E 334 : BRB SETEOF ; bad syntax - give eof error
00A0 335 :+++
00A0 336 : have found $deck /dollars :
00A0 337 : scan for end-of-data string value
00A0 338 :--+
00A0 339 : GOT_ARG:
00A0 340 : DCL_SCAN <','>,EQUAL=GOT_QUOTE ; scan for quoted string
00A0 341 :--+
00A0 342 : BEQL SETEOD ; branch if nothing else
00A0 343 :+++
00A0 344 : have an unquoted, non-null end-of-data string value described by r0,r1
00A0 345 : copy characters to end of data string up to first blank, tab or "!!",
00A0 346 : converting them to upper case.
00A0 347 :--+
00A0 348 :--+
00A0 349 : DCL_SCAN <','>,EQUAL=GOT_QUOTE ; scan for quoted string
00A0 350 :--+
00A6 351 : BEQL SETEOD ; branch if nothing else
00A8 352 :+++
00A8 353 : have found $deck /dollars : "
00A8 354 :--+
00A8 355 : scan for closing quote, moving characters to end of data string.
00A8 356 : process such that successive double quotes cause a single double quote
00A8 357 : to be entered into the end of data string.
00A8 358 :--+
00A8 359 :--+
00A8 360 :--+
00A8 361 :--+
52 01 D0 00A8 362 : MOVL #1,R2 ; flag unquoted string value
02 11 00AB 363 : BRB UNQUOTED
00AD 364 :--+
00AD 365 :+++
00AD 366 : have found $deck /dollars : "
00AD 367 :--+
00AD 368 : scan for closing quote, moving characters to end of data string.
00AD 369 : process such that successive double quotes cause a single double quote
00AD 370 : to be entered into the end of data string.
00AD 371 :--+
00AD 372 :--+
00AD 373 :--+
00AD 374 :--+
00AD 375 : GOT_QUOTE:
52 D4 00AD 376 : CLRL R2 ; flag quoted string
00AF 377 : UNQUOTED:
00AF 378 : CLRL R5 ; build string count here
00B1 379 : MOVAB @#PIO$GT-ENDSTR,R4 ; addr of eod string length
00B8 380 : MOVAB 1(R4),R3 ; addr of eod string text

```

83	03	11	00BC	381	BRB	20\$		
	81	90	00BE	382	10\$:	MOVB	(R1)+,(R3)+	; go process characters
	50	D7	00C1	383	20\$:	DECL	R0	; copy char to eod string
	4C	19	00C3	384		BLSS	60\$	; any more characters?
10	52	E8	00C5	385		BLBS	R2,45\$	; branch if not
22	61	91	00C8	386		CMPB	(R1),#^A//	; branch if unquoted string
	32	13	00CB	387		BEQL	50\$	; matching quote?
ED	55	0F	F3	388	30\$:	AOBLEQ	#EODSTR_MAXLEN,R5,10\$	; branch if yes
								; count char. and branch if ok
				389				
				390				;++
				391				
				392				exceeded max character count. reset eod match string to single '\$'.
				393				
				394				--
01	A4	24	90	395				
	FF6D	31	00D1	396	40\$:	MOVB	#^A/\$/,1(R4)	; restore match string
				397		BRW	SETEOF	; go give error
				398				
				399				;++
				400				
				401				unquoted string
				402				
				403				move character to e-o-d string unless it's blank, tab, or "!"
				404				and convert to upper case.
				405				
				406				--
				407				
20	61	91	00D8	408	45\$:	CMPB	(R1),#^A/ /	; space?
	2D	13	00DB	409		BEQL	55\$	; branch if yes
09	61	91	00DD	410		CMPB	(R1),#TAB	; tab?
	28	13	00EO	411		BEQL	55\$	; branch if yes
21	61	91	00E2	412		CMPB	(R1),#^A//	; "!"?
	23	13	00E5	413		BEQL	55\$	; branch if yes
61	8F	61	00E7	414		CMPB	(R1),#LOWERCASE_A	; lower case char?
	E0	1F	00EB	415		BLSSU	30\$	; branch if not
7A	8F	61	00ED	416		CMPB	(R1),#LOWERCASE_Z	; well, is it?
	DA	1A	00F1	417		BGTRU	30\$	; branch if not
02	55	0F	F3	418		AOBLEQ	#EODSTR_MAXLEN,R5,48\$	; count char. & branch if ok
		D8	11	419		BRB	40\$	; go process eod length error
83	81	20	83	420	48\$:	SUBB3	#LOWERCASE_A-^A/A/,,(R1)+,(R3)+; convert to upper case	
		C2	11	421		BRB	20\$	; go get next char.

```

00FF 423
00FF 424 :++
00FF 425
00FF 426 found a double quote character while processing quoted string
00FF 427
00FF 428 check next char. for double quote and include only one if found
00FF 429
00FF 430 :--
00FF 431
22 50 D7 00FF 432 50$: DECL R0 ; any more characters?
0E 19 0101 433 BLSS 60$ ; branch if not
51 D6 0103 434 INCL R1 ; point to next char
61 91 0105 435 CMPB (R1),#^A// ; is it another " ?
C3 13 0108 436 BEQL 30$ ; branch if yes
010A 437
010A 438 :++
010A 439
010A 440 have completed string value, but remaining string is non-null.
010A 441
010A 442 give an error if anything other than blanks, tabs, or comment.
010A 443
010A 444 :--
010A 445
50 D6 010A 446 55$: INCL R0 ; restore character count
0077 30 010C 447 BSBW BLNK_SKIP ; skip blanks and tabs
C0 12 010F 448 BNEQ 40$ ; branch if other than comment
0111 449
0111 450 :++
0111 451
0111 452 end of data string set up o.k.
0111 453
0111 454 store length and go skip record.
0111 455
0111 456 :--
0111 457
55 95 0111 458 60$: TSTB R5 ; any chars processed?
03 12 0113 459 BNEQ 70$ ; branch if none
FF65 31 0115 460 BRW SETEOD ; store count
64 55 90 0118 461 70$: MOVB R5,(R4) ; go skip record
FF66 31 011B 462 BRW SET_SKIP
011E 463

```

```

011E 465      .SBTTL  DCL_SCAN SUBROUTINE
011E 466
011E 467 :++
011E 468
011E 469      dcl_scan subroutine to scan for next token and compare it to one
011E 470      being searched for. case is not significant for the compare.
011E 471      any initial blanks or tabs are skipped over.
011E 472
011E 473      if the strings match, the return is made to the address specified in the
011E 474      "equal" input argument, otherwise return is made in line.
011E 475      note that the dcl_scan macro is used to set up the in-line argument list
011E 476
011E 477      in the case of strings other than length 1, any characters following
011E 478      the matched characters and before the next terminator are considered
011E 479      to be part of the token and are also skipped in setting the remaining
011E 480      string descriptor, as are any trailing blanks or tabs.
011E 481
011E 482      inputs:
011E 483
011E 484      r0      remaining string length
011E 485      r1      remaining string start address
011E 486      (sp)    counted, upper-case string to match
011E 487      (sp)+count  branch byte offset for equal compare
011E 488
011E 489      outputs:
011E 490
011E 491      r0      length of remaining string (past token and possible
011E 492      trailing blanks if matched)
011E 493      r1      address of remaining string
011E 494      r2-r5,ap  destroyed
011E 495
011E 496      notes:
011E 497
011E 498      1. if no match, r0 & r1 will be updated to point past any initial
011E 499      spaces and or tabs
011E 500      2. r0 will be set to 0 on return if no string or only a comment remains
011E 501      3. z-bit will be set based on r0
011E 502
011E 503 :--
011E 504
011E 505      TRMLST: .ASCII  \ !=:/\<TAB>
0124 506      TLSTSZ=-TRMLST
0124 507
0124 508      DCL_SCAN:
0124 509      MOVL    (SP),R5      ; get addr of counted ascii
0127 510
0127 511 : match string
0127 512
0127 513
0127 514
0127 515      MOVZBL  (R5)+,R4      ; get length of string
012A 516
012A 517
012A 518 : (r5 now points to string)
012A 519
012A 520
012A 521      ADDL2  R4,(SP)      ; bump return address ...

```

6E	D6	012D	522	INCL	(SP)	; ... to point past ascii string
		012F	523			
		012F	524	++		
		012F	525			
		012F	526	skip initial spaces and tabs		
		012F	527			
		012F	528	--		
		012F	529			
55	10	012F	530	BSBB	BLNK_SKIP	; skip tabs and blanks
4E	13	0131	531	BEQL	NULL_STRING	; branch if nothing left
		0133	532			
		0133	533	++		
		0133	534			
		0133	535	the string described by r0, r1 is of non-zero length and does not begin		
		0133	536	with tab, space, or ":"		
		0133	537			
		0133	538	see if it matches the scan string (described by r4 & r5)		
		0133	539			
		0133	540	--		
		0133	541			
52	50	7D	0133	542	MOVQ R0,R2	; save remaining len and addr
50	54	C2	0136	543	SUBL2 R4,R0	at least match count long?
	43	19	0139	544	BLSS NOMATCH	branch if not
	54	DD	013B	545	PUSHL R4	save match string count
61	5C	81	013D	546	10\$: MOVB (R1)+,AP	get next byte
	8F	5C	91	0140	CMPB AP,#LOWERCASE_A	lower case?
	09	1F	0144	547	BLSSU 20\$	branch if not
7A	8F	5C	91	0146	548 CMPB AP,#LOWERCASE_Z	well, is it?
	03	1A	014A	549	BGTRU 20\$	branch if not
	5C	20	014C	550	SUBB2 #LOWERCASE_A - <^A/A>,AP; convert to upper case	
	85	5C	91	014F	551 CMPB AP,(R5)+	match?
	28	12	0152	552	20\$: BNEQ UNEQUAL	branch if not
E6	54	F5	0154	553	SOBGTR R4,10\$	loop

```

0157 556
0157 557 :++
0157 558
0157 559 strings are equal
0157 560
0157 561 if match count is not = 1, scan to end of token and then to start of next token
0157 562 (end of token is indicated by space, tab, !, /, =, or :)
0157 563
0157 564 in any case, take the "equal=" exit
0157 565
0157 566 :++
0157 567 ;--
0157 568

      8E  D7 0157 569      DECL   (SP)+      ; was match count = 1?
      18  13 0159 570      BEQL   60$      ; branch if yes
      52  50 7D 015B 571      MOVQ   R0,R2      ; save remaining descriptor
      07  11 015E 572      BRB    40$      ; is character a delimiter?
      B9 AF 06 83 3A 0160 573 30$:      LOC C (R3)+,#TLSTSZ,TRMLST      ; branch if yes
      03  12 0165 574      BNEQ   50$      ; loop if more characters
      F6  52 F4 0167 575 40$:      SOBGEQ R2,30$      ; don't count terminator
      52  D6 016A 576 50$:      INCL    R2      ; or point past it
      53  D7 016C 577      DECL    R3      ; descriptor to right regs
      50  52 7D 016E 578      MOVQ   R2,R0      ; go skip tabs and blanks
      13  10 0171 579      BSB B BLNK SKIP      ; pick up "equal" branch offset
      52  00 BE 98 0173 580 60$:      CVTBL  @(SP),R2      ; add in offset to return pc
      6E  52 C0 0177 581      ADDL2  R2,(SP)
      07  11 017A 582      BRB    SCAN_XIT      ; add in offset to return pc

      017C 583
      017C 584 :++
      017C 585
      017C 586 the input string didn't contain the match string
      017C 587
      017C 588 leave r0,r1 describing any remaining string and take in-line (non-equal) return
      017C 589
      017C 590 ;--
      017C 591
      017C 592 UNEQUAL:
      8E  D5 017C 593      TSTL   (SP)+      ; pop saved match count
      50  52 7D 017E 594      NOMATCH:      ; restore save descriptor
      017E 595      MOVQ   R2,R0      ; skip past "equal" return offset
      6E  D6 0181 596      NULL_STRING:      ; set z bit according to r0
      0181 597      INCL   (SP)
      50  D5 0183 598      SCAN_XIT:      ; RSB
      05  0185 599      TSTL   R0
      600

```

```

0186 602
0186 603 :++
0186 604
0186 605      blink_skip subroutine to skip past blanks and tabs, up to possible comment
0186 606      or end of input string
0186 607
0186 608      inputs:
0186 609
0186 610      r0      input string length
0186 611      r1      input string address
0186 612
0186 613      outputs:
0186 614
0186 615      r0      remaining string length after blanks and tabs skipped
0186 616      (if only a comment left, r0 will be set to zero)
0186 617      r1      remaining string address
0186 618      z-bit   set if no more input (other than comment), else clear
0186 619
0186 620 :--
0186 621
0186 622 BLNK_SKIP:
61 50 20 3B 0186 623 10$:  SKPC  #^A/ .,R0,(R1)      ; skip spaces
61 12 13 018A 624  BEQL  30$               ; branch if nothing but spaces
61 09 91 018C 625  CMPB  #TAB,(R1)          ; is char tab?
61 06 12 018F 626  BNEQ  20$               ; branch if not (done)
50 D7 0191 627  DECL   R0                 ; yes - decrement count
51 D6 0193 628  INCL   R1                 ; skip tab
EF 11 0195 629  BRB   10$               ; and continue skipping
61 21 91 0197 630 20$:  CMPB  #^A/!/, (R1)    ; do we have a comment?
02 12 019A 631  BNEQ  30$               ; branch if not
50 D4 019C 632  CLRL   R0                 ; yes - say end of input
05 019E 633 30$:  RSB                ; return with z-bit set if no
019F 634
019F 635
019F 636 : more input
019F 637
019F 638
019F 639
019F 640      .END

```

\$\$PSECT EP  
 \$\$RMSTEST  
 \$\$RMS\_PBUGCHK  
 \$\$RMS\_TBUGCHK  
 \$\$RMS\_UMODE  
 BLNK SKIP  
 DCL SCAN  
 END\_OF DATA  
 END\_OF\_DATA1  
 EODT  
 EODSTR MAXLEN  
 EOD\_MATCH  
 GOT\_ARG  
 GOT\_DECK  
 GOT\_DOLLARS  
 GOT\_QUOTE  
 IRBSL\_NRP VBN  
 IRBSL\_RP VBN  
 IRBSV\_FIND  
 IRBSV\_PPF\_EOF  
 IRBSV\_PPF\_FNDSV  
 IRBSV\_PPF\_SKIP  
 LOWERCASE\_A  
 LOWERCASE\_Z  
 NOMATCH  
 NULL\_STRING  
 PIOSGT\_ENDSTR  
 PIOSGW\_STATUS  
 PIOSV\_EOD  
 RM\$INPUT\_SCAN  
 RMSS\_EOF  
 SCAN\_XIT  
 SCNR5B  
 SCNXIT  
 SETEOD  
 SETEOF  
 SET\_FIND  
 SET\_SKIP  
 SUCXIT  
 SUC\_BR  
 TAB  
 TLSTSZ  
 TRMLST  
 UNEQUAL  
 UNQUOTED

= 00000000  
 = 0000001A  
 = 00000010  
 = 00000008  
 = 00000004  
 00000186 R 01  
 00000124 R 01  
 00000051 R 01  
 0000005C R 01  
 00000067 R 01  
 = 0000000F  
 0000001D R 01  
 000000A0 R 01  
 00000071 R 01  
 00000090 R 01  
 000000AD R 01  
 = 00000040  
 = 00000048  
 = 00000029  
 = 0000002E  
 = 00000030  
 = 0000002F  
 = 00000061  
 = 0000007A  
 0000017E R 01  
 00000181 R 01  
 \*\*\*\*\* X 01  
 \*\*\*\*\* X 01  
 = 00000001  
 00000000 RG 01  
 = 0001827A  
 00000183 R 01  
 0000001C R 01  
 0000001A R 01  
 0000007D R 01  
 00000045 R 01  
 0000006B R 01  
 00000084 R 01  
 00000017 R 01  
 0000006F R 01  
 = 00000009  
 = 00000006  
 0000011E R 01  
 0000017C R 01  
 000000AF R 01

+-----+  
 ! Psect synopsis !  
 +-----+

## PSECT name

## Allocation

## PSECT No.

## Attributes

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
RMSRMS1	0000019F ( 415.)	01 ( 1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
SABSS	00000000 ( 0.)	02 ( 2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

+-----+  
! Performance indicators !  
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.09	00:00:00.93
Command processing	132	00:00:00.70	00:00:05.80
Pass 1	271	00:00:07.54	00:00:22.25
Symbol table sort	0	00:00:00.93	00:00:02.03
Pass 2	122	00:00:02.08	00:00:08.15
Symbol table output	7	00:00:00.06	00:00:00.55
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	571	00:00:11.42	00:00:39.74

The working set limit was 1200 pages.

42967 bytes (84 pages) of virtual memory were used to buffer the intermediate code.

There were 40 pages of symbol table space allocated to hold 747 non-local and 28 local symbols.

640 source lines were read in Pass 1, producing 14 object records in Pass 2.

20 pages of virtual memory were used to define 19 macros.

+-----+  
! Macro library statistics !  
+-----+

Macro library name	Macros defined
\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	10
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	0
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	14

832 GETS were required to define 14 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:RM1INPSCN/OBJ=OBJ\$:RM1INPSCN MSRC\$:RM1INPSCN/UPDATE=(ENH\$:RM1INPSCN)+EXECMLS\$/LIB+LIB\$:RMS/LIB

0321 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

RMICONN  
LIS

RMIGET  
LIS

RMIINPSN  
LIS

RMLDISCON  
LIS

RMIGETINT  
LIS

RMICREATE  
LIS

RMIJOURNL  
LIS